# Syntree

A library for producing syntax tree graphs.

Version 0.1.0 2025-02-23 MIT

## Table of Contents

# Part I Usage

## I.1 Importing the Package

```
#import "@preview/syntree:0.1.0"
```

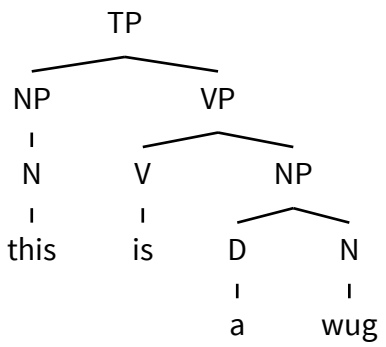
## I.2 Building a Syntax Tree

Each node in the tree is an array of a label and a child node array, with the exception of terminal nodes, which contain content instead.

Be aware that the trailing commas after lone children are *not* optional, as otherwise they cause `("NP", (("N", "this")))` to parse as `("NP", ("N", "this"))`.

```
#let tree = ("TP", (
  ("NP", (
    ("N", "this"),
  )),
  ("VP", (
    ("V", "is"),
    ("NP", (
      ("D", "a"),
      ("N", "wug"),
    )),
  )),
))

#syntree.tree(tree, min-gap-x: 1)
```



# Part II Arguments

```
#tree(
  (node),
  (hug-bottom): false,
  (min-space-y): 1.0,
  (min-gap-x): 0.3,
  (min-slope): 0,
  (full-step-y): false,
  (label-alignment): "middle"
) → content
```

A function which takes a hierarchy of syntax elements and displays a tree of them.

Argument

(node) `array`

The root node of the syntax tree. See the examples for its structure.

Argument

(hug-bottom): false `boolean`

Draws all words at the bottom in a line, with stems extending down from their parent. It does not function correctly with a non-default min-slope.

Argument

(min-space-y): 1.0 `float`

The vertical separation of levels of the tree. It may be higher than this if min-slope is set.

Argument

(min-gap-x): 0.3 `float`

The minimum horizontal gap between terminal nodes, which applies even if they are not vertically aligned.

Argument

(min-slope): 0 `float`

Sets an approximate lower bound for the slope of the connecting lines. Setting it to a non-zero value will change the vertical spacing of the graph. 0.2 has worked well for me, but this can be any number.

Argument

(full-step-y): false `boolean`

To space tree layers by increments of min-space-y when using a min-slope. This prevents layers from getting “out of sync” and having words that are slightly out of alignment with each other.

Argument

⟨label-alignment⟩: "middle" string

This accepts one of three strings: “middle”, “average”, or “smart”.

Middle places a parent label exactly between its outer children’s labels (the ones furthest to the left and right)

Average places it at the average of its children’s horizontal positions, which can look better when there are many clumped close together and only one off to the side.

Smart acts like average, but if there are three children and the second label is sufficiently close to the middle, the parent will “snap” to it.

# Part III Examples

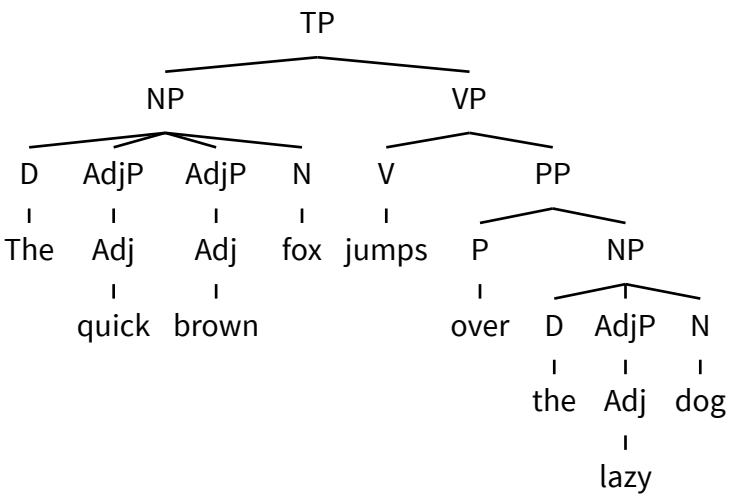
Note that I am not a linguist, so these analyses may be wrong.

## III.1 The Quick Brown Fox

```
#let tree = ("TP", (
  ("NP", (
    ("D", "The"),
    ("AdjP", (
      ("Adj", ("quick")),
    )),
    ("AdjP", (
      ("Adj", ("brown")),
    )),
    ("N", "fox"),
  )),
  ("VP", (
    ("V", "jumps"),
    ("PP", (
      ("P", "over"),
      ("NP", (
        ("D", "the"),
        ("AdjP", (
          ("Adj", "lazy"),
        )),
        ("N", "dog"),
      )),
    )),
  )),
))

#syntree.tree(tree, label-alignment: "smart")
```

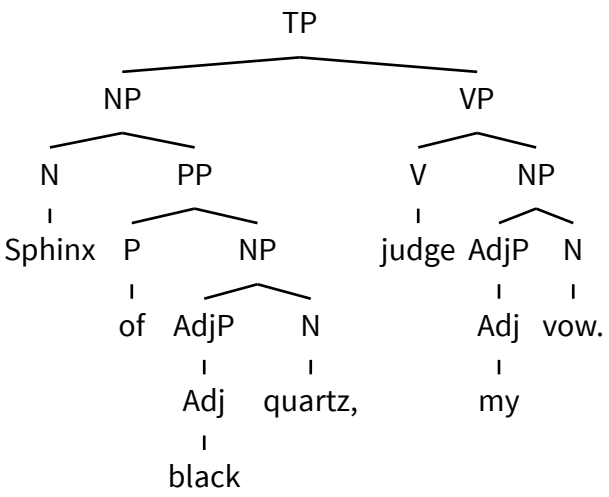
---



## III.2 Sphinx of Black Quartz

```
#let tree = ("TP", (
  ("NP", (
    ("N", "Sphinx"),
    ("PP", (
      ("P", "of"),
      ("NP", (
        ("AdjP", (
          ("Adj", "black"),
        )),
        ("N", "quartz,")
      )),
    )),
  )),
  ("VP", (
    ("V", "judge"),
    ("NP", (
      ("AdjP", (
        ("Adj", "my"),
      )),
      ("N", "vow."),
    )),
  )),
))

#syntree.tree(tree)
```



## III.3 The Move from a Structuralist Account

```
#let tree = ("TP",(("NP",(("D","The"),("N","move"),("PP",(("P","from"),("NP",
(("D","a"),("AdjP",(("Adj","structuralist"),)),("N","account"),("PP",(("P","in"),
("CP",(("C","which"),("TP",(("NP",(("N","capital"),)),("T","is"),("VP",
(("V","understood"),("TP",(("T","to"),("VP",(("V","structure"),("NP",(("AdjP",
(("Adj","social"),)),("N","relations"),)),("PP",(("P","in"),("NP",(("AdjP",(("AdvP",
(("Adv","relatively"),)),("Adj","homologous"),)),
("N","ways"),)),)),)),)),)),)),)),)),)),)),("PP",(("P","to"),("NP",(("D","a"),
("N","view"),("PP",(("P","of"),("NP",(("N","hegemony"),)),)),("PP",(("P","in"),
("CP",(("C","which"),("TP",(("NP",(("AdjP",(("Adj","power"),)),("N","relations"),)),
("VP",(("V","are"),("AdjP",(("Adj","subject"),("PP",(("P","to"),("NP",
(("N","repetition"),("N","convergence"),("Conj","and"),
("N","rearticulation"),)),)),)),)),)),)),)),)),)),)),("VP",(("V","brought"),("NP",
(("D","the"),("N","question"),("PP",(("P","of"),("NP",(("N","temporality"),)),)),)),
("PP",(("P","into"),("NP",(("D","the"),("N","thinking"),("PP",(("P","of"),("NP",
(("N","structure"),)),)),)),)),)),("Conj","and"),("VP",(("V","marked"),("NP",
(("D","a"),("N","shift"),)),("PP",(("P","from"),("NP",(("D","a"),("N","form"),("PP",
(("P","of"),("NP",(("AdjP",(("Adj","Althusserian"),)),("N","theory"),)),)),)),("CP",
(("C","that"),("TP",(("VP",(("V","takes"),("NP",(("AdjP",(("Adj","structural"),)),
("N","totalities"),)),("PP",(("P","as"),("NP",(("AdjP",(("Adj","theoretical"),)),
("N","objects"),)),)),)),)),)),)),)),("PP",(("P","to"),("NP",(("N","one"),("PP",
(("P","in"),("CP",(("C","which"),("TP",(("NP",(("D","the"),("N","insights"),("PP",
(("P","into"),("NP",(("D","the"),("AdjP",(("Adj","contingent"),)),
("N","possibility"),("PP",(("P","of"),("NP",(("N","structure"),)),)),)),)),)),)),("VP",
(("V","inaugurate"),("NP",(("D","a"),("AdjP",(("Adj","renewed"),)),
("N","conception"),("PP",(("P","of"),("NP",(("N","hegemony"),)),)),("PP",
(("P","as"),("AdjP",(("Adj","bound"),("AP",(("Adv","up"),)),("PP",(("P","with"),
("NP",(("D","the"),("AdjP",(("Adj","contingent"),)),("N","sites"),("Conj","and"),
("N","strategies"),("PP",(("P","of"),("NP",(("D","the"),("N","rearticulation"),
("PP",(("P","of"),("NP",
(("N","power"),)),)),)),)),)),)),)),)),)),)),)),)),)),)),))

#scale(11%, reflow: true, syntree.tree(tree, label-alignment: "average", min-space-
y: 2))
```

---